Ingenieurbüro für Echtzeitprogrammierung

# RTP1

**Dok-Rev. 1.6 vom 04.06.2019**
**Hardware-Rev. 1.7 vom 01.10.2018**

# Inhaltsverzeichnis

Revisionsliste:

| Rev. | Datum | Na. | Änderung |
|---|---|---|---|
| 1.0 | 23.10.2017 | Ko | Erstellung |
| 1.1 | 14.12.2017 | Ko | Beschreibung analoge Eingänge korrigiert |
| 1.2 | 08.06.2018 | Ko | Dig. Eingänge korrigiert |
| 1.3 | 30.12.2018 | Ko | Analoge Ausgänge GND bei ST7 korrigiert |
| 1.4 | 05.12.2018 | Ko | Beschreibung erweitert |
| 1.5 | 27.05.2019 | Kr | SPI-IO Geräte / Terminalemulation |
| 1.6 | 04.06.2019 | Ko | Ergänzungen |
| | | | |

# 1 Allgemeine Hinweise

## 1.1 Handhabung

1. Lesen Sie bitte zuerst sorgfältig diese Dokumentation bevor Sie die Hardware auspacken und einschalten. Sie sparen Zeit und vermeiden Probleme.
2. Beachten Sie bitte die Vorsichtsmaßnahmen bei der Handhabung elektrostatisch gefährdeter Hardware.
3. Wenn die Hardware Batterien enthält, legen Sie sie nicht auf elektrisch leitfähige Unterlagen. Die Batterie könnte kurzgeschlossen werden und Schäden verursachen.
4. Achten Sie bitte darauf, daß der spezifizierte Temperaturbereich nicht verlassen wird.

## 1.2 Installation

1. Überprüfen Sie, ob alle Jumper entsprechend Ihrer Anwendung gesetzt sind.
2. Schalten Sie die Spannungsversorgung der externen Anschlüsse ab, bevor Sie eine Verbindung herstellen.
3. Wenn Sie sicher sind, daß alle Verbindungen korrekt installiert sind, schalten Sie die Spannungsversorgung ein.

## 1.3 Erklärung

Wir behalten uns das Recht vor, Änderungen, die einer Verbesserung der Schaltung oder des Produktes dienen, ohne besondere Hinweise vorzunehmen. Trotz sorgfältiger Kontrolle kann für die Richtigkeit der hier gegebenen Daten, Schaltpläne, Programme und Beschreibungen keine Haftung übernommen werden. Die Eignung des Produktes für einen bestimmten Einsatzzweck wird nicht zugesichert.

## 1.4 Reparaturen

Sollte das Produkt defekt sein, so senden Sie es bitte frei in geeigneter Verpackung mit folgender Beschreibung an uns zurück:

- Fehlerbeschreibung
- Trat der Fehler nur unter bestimmten Bedingungen auf?
- Was war angeschlossen?
- Wie sahen die angeschlossenen Signale aus?
- Garantiereparatur oder nicht?

# 2 Technische Daten

## 2.1 Umgebungsbedingungen

| | |
|---|---|
| Umgebungstemperatur (Betrieb) | 0-50° C |
| Umgebungstemperatur (Lagerung) | -20-85° C |
| rel. Luftfeuchte | max. 95%, nicht kondensierend |
| Höhe | -300m bis +3000m |

## 2.2 Abmessungen

200 x 110 mm

## 2.3 Anschlüsse

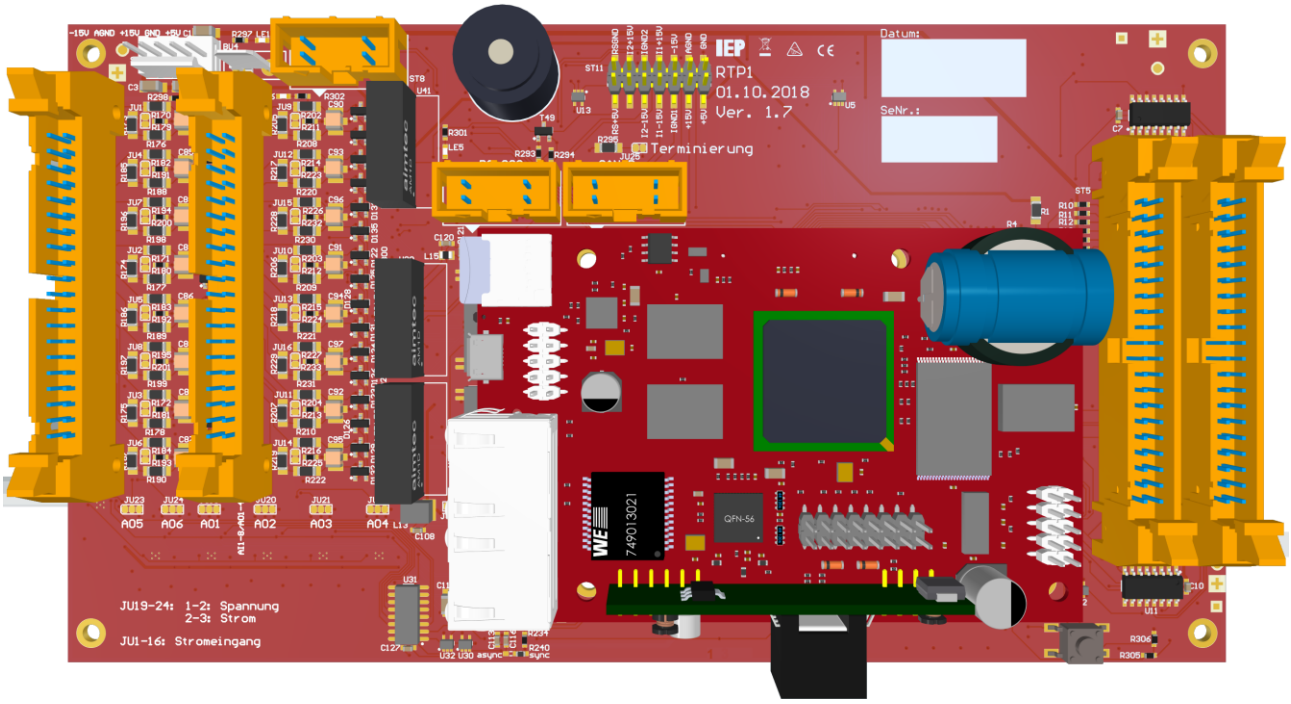
| | |
|---|---|
| Versorgungsspannung: | 5 Volt DC ±5%, max. 3 A incl. Ausgänge<br>±15 Volt DC ±5%, max. 0,3 A |
| Eingänge: | 16 digitale Eingänge, Schaltschwelle ca. bei 3,1 V, Strom 1 mA, Tiefpass 48 Hz<br>16 analoge differenzielle Eingänge, 0..10 V oder 0..20 mA (Jumper), 24 Bit Auflösung, Eingangsfilter 158 Hz |
| Ausgänge: | 16 digitale Ausgänge, 5 V / 100 mA je Ausgang, Summenstrom aller Ausgänge max. 1,5 A<br>8 analoge Ausgänge, davon 2 einzeln galvanisch getrennt, 12 Bit Auflösung, 0..10V, 6 Stück umschaltbar 4..20 mA strombegrenzt |
| Ein-/Ausgänge: | 32 gemischt nutzbare digitale Kanäle, s.o. |
| Serielle Schnittstellen: | 3 serielle 5-Draht Schnittstellen, eine galvanisch getrennt, Baudrate bis 115 KBaud |
| CANBus: | bis 1 MBit |
| Displayanschluß: | 7" Farb-TFT, 800x480 Pixel, Touchscreen |
| Ethernet: | 2x RJ45 10/100 MBit |
| µSD-Karte / USB: | Massenspeicher |
| Signalgeber: | 1,9 KHz; 85 dB(A) / 10cm |

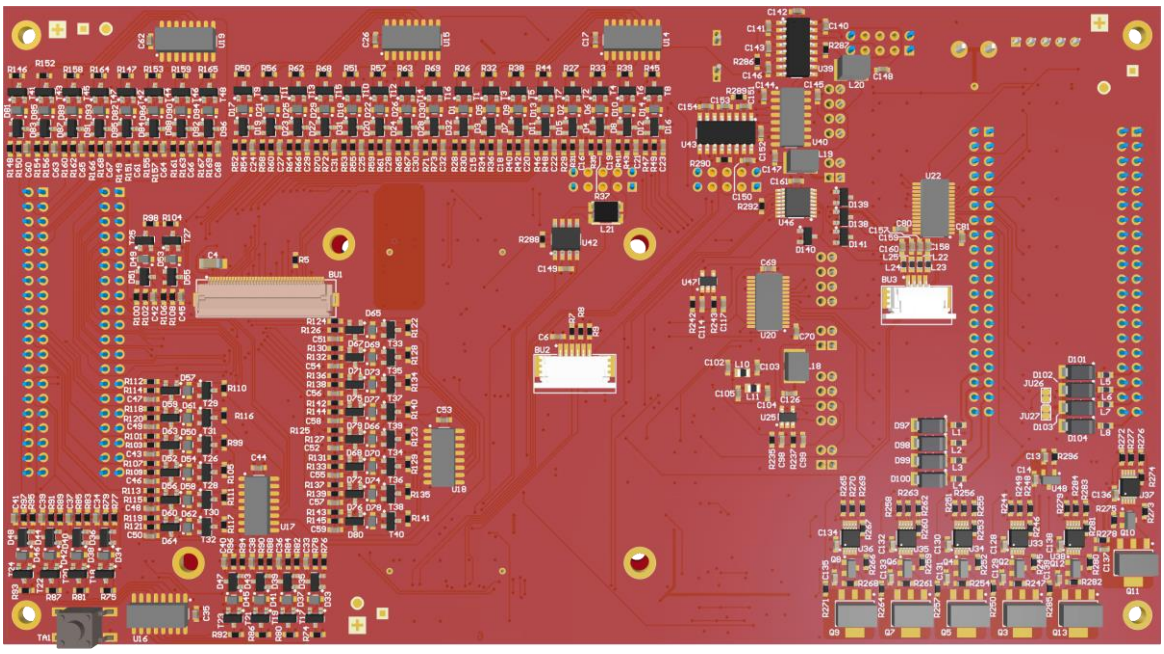
# 3 Inbetriebnahme

## 3.1 Einbau

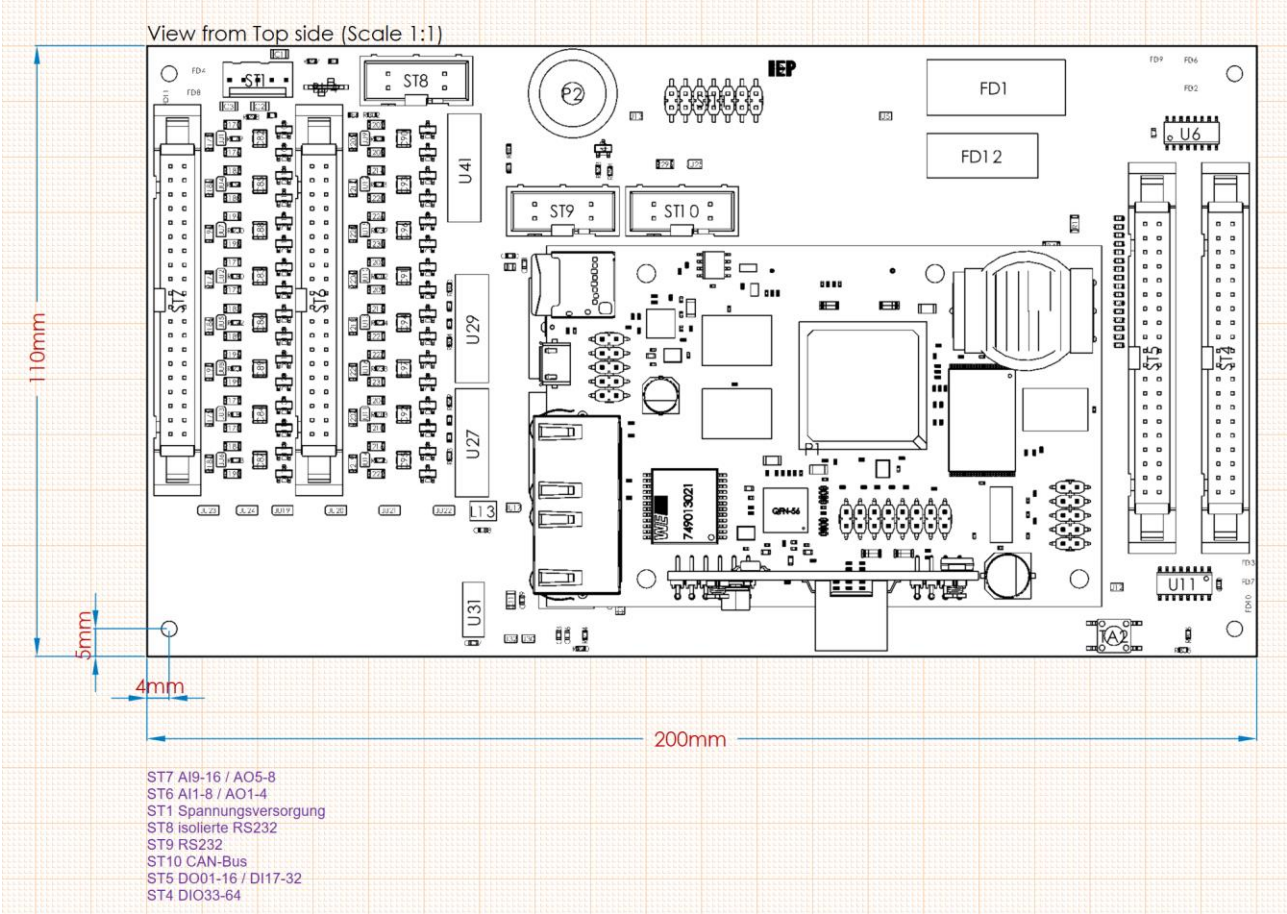
Der RTP1 ist zum Einbau in Schaltschränke oder ähnliche EMV-dichte Gehäuse bestimmt. Die Verkabelung ist EMV-gerecht mit abgeschirmten Kabeln durchzuführen.

## 3.2 Ansichten



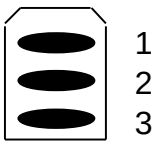



## 3.3 Einbaumaße




## 3.4 Beschreibung der Jumper

Die Löt-Jumper werden folgendermaßen gezählt:




**JU1 – JU16**:

Beim Schliessen dieser Jumper wird der entsprechende AI-Kanal auf 0..20 mA Eingangsbereich gestellt. Die Zuordnung ist folgendermaßen:

AI1-JU1, AI4-JU2, AI7-JU3
AI2-JU4, AI5-JU5, AI8-JU6
AI3-JU7, AI6-JU8

AI9-JU9, AI12-JU10, AI15-JU11
AI10-JU12, AI13-JU13, AI16-JU14
AI11-JU15, AI14-JU16

**JU17 / JU18**:

Wenn keine galvanisch getrennten Analogausgänge benötigt werden, kann die entsprechende Bestückung weg gelassen werden (insbesonders U25/U47). Über den geschlossenen Jumper wird der jeweilige Ausgang (AO7/AO8) versorgt.

**JU26 / JU27:**

Hiermit wird die galvanische Trennung des jeweiligen analogen Ausgangs aufgehoben, indem die Massen verbunden werden. JU26 verbindet die Massen von AO7, JU27 von AO8.

**JU19 – JU24:**

Diese Jumper schalten die analogen Ausgänge zwischen Spannungs- oder Stromausgang um.

| JU19 – JU24 | AO1 – AO6 |
|---|---|
| 1 – 2 | Spannung |
| 2 – 3 | Strom |

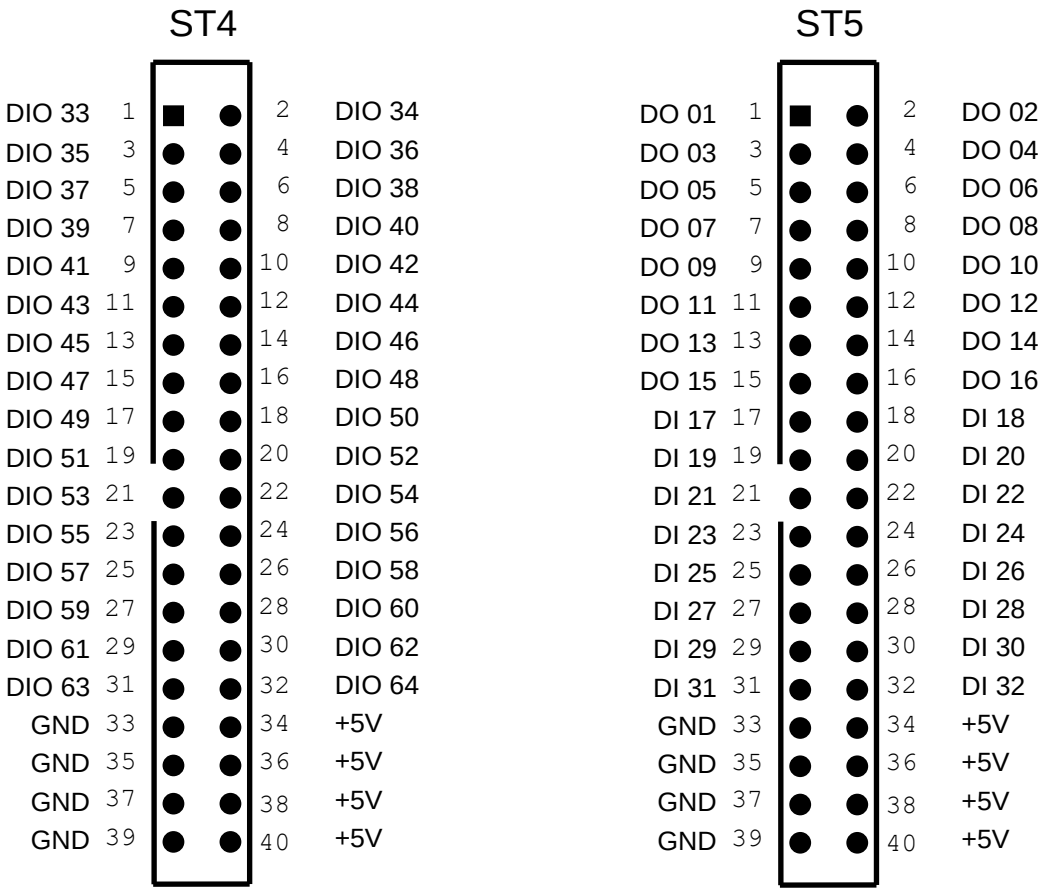
**JU25**

Dieser Jumper schaltet den Abschlußwiderstand des CAN-Busses ein, wenn er geschlossen ist.

# 4 Hardwarebeschreibung

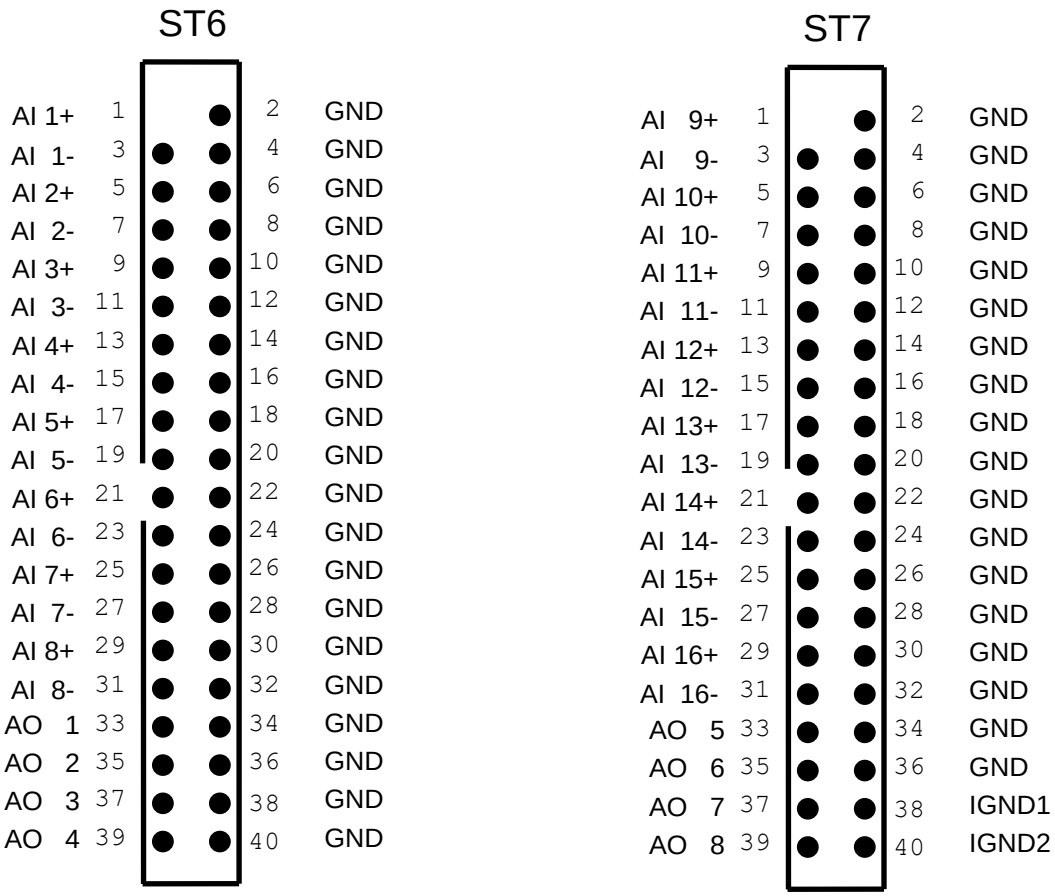
## 4.1 Steckerbelegungen

### 4.1.1 DIO 1-64




DIO33-64 können einzeln als Ein- oder Ausgang genutzt werden. DO01-16 sind nur Ausgänge, DI17-32 nur Eingänge.

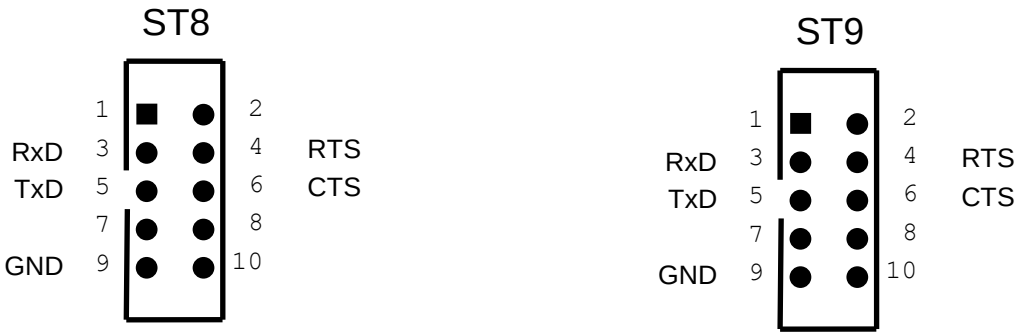
### 4.1.2 AIO 1-16




AO7/8 können als galvanisch getrennte analoge Ausgänge ausgeführt werden!

### 4.1.3 Schnittstellen

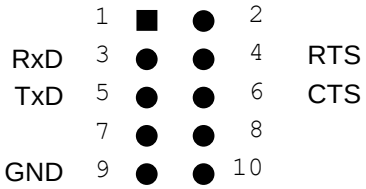
#### *4.1.3.1 RS-232*




ST8 ist galvanisch vom Rest der Schaltung getrennt

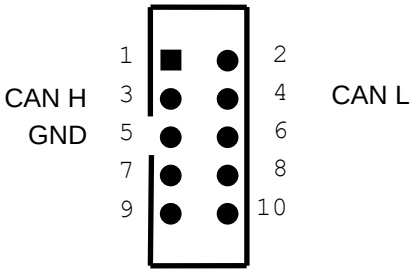
Eine weitere serielle Schnittstelle steht auf dem CORE5125 zur Verfügung:

ST5



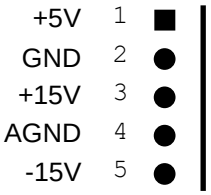

#### *4.1.3.2 CAN-Bus*

ST10




### 4.1.4 Spannungsversorgung

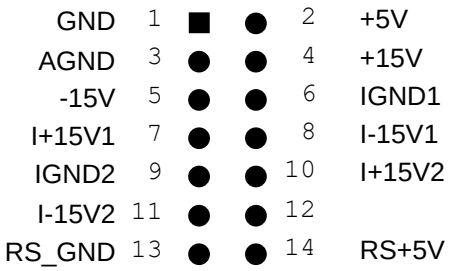
ST1




GND und AGND sind intern verbunden!

### 4.1.5 Diagnoseanschluß

ST11



## 4.2 LED

Die Spannungen werden über folgende LED angezeigt:

| LED | Spannung | Bedeutung |
|---|---|---|
| LE1 | +5V | Versorgung CORE5125, dig. I/O |
| LE2 | +15V | Analoge Ein-/Ausgänge |
| LE3 | I+15V1 | Galvanische Trennung AO7 |
| LE4 | I+15V2 | Galvanische Trennung AO8 |
| LE5 | RS+5V | Serielle Schnittstelle ST8 |
| LE6 | -15V | Analoge Ein-/Ausgänge |
| LE7 | I-15V1 | Galvanische Trennung AO7 |
| LE8 | I-15V2 | Galvanische Trennung AO8 |

## 4.3 Reset-Taster

Auf beiden Seiten der Platine kann ein Reset-Taster (TA1/2) bestückt werden. Beim Drücken wird ein Reset ausgelöst.

## 4.4 Signalgeber

Mit dem Signalgeber P2 kann ein lauter Pipton ausgelöst werden.

## 4.5 Display

Das NTP1 ist für den Anschluß eines 7"-Displays mit der Auflösung von 800x480 Pixel vorgesehen. Es werden sowohl resistive als auch kapazitive Touchscreens unterstützt.

Das Backlight wird mit einem konstanten Strom von ca. 180 mA versorgt und kann über eine PWM gedimmt werden.

### 4.5.1 Resistiver Touch

| Pin | Belegung |
|---|---|
| 1 | X-Left |
| 2 | Y-Down |
| 3 | X-Right |
| 4 | Y-Up |

### 4.5.2 Kapazitiver Touch

| PIN | Belegung |
|---|---|
| 1 | +3,3 Volt |
| 2 | GND |
| 3 | I²C SCL |
| 4 | I²C SDA |
| 5 | \IRQ |
| 6 | \Reset |

### 4.5.3 Display Anschluß

| PIN | Belegung |
|---|---|
| 1 | Backlight - |
| 2 | Backlight + |
| 3 | GND |
| 4 | +3,3 Volt |
| 5 | Rot LSB |
| 6-11 | … |
| 12 | Rot MSB |
| 13 | Grün LSB |
| 14-19 | … |
| 20 | Grün MSB |
| 21 | Blau LSB |
| 22-27 | … |
| 28 | Blau MSB |
| 29 | GND |
| 30 | CLK |
| 31 | On/Off |
| 32 | HSYNC |
| 33 | VSYNC |
| 34 | DE |
| 35 | NC |
| 36 | GND |
| 37 | NC |
| 38 | NC |
| 39 | NC |
| 40 | NC |

# 5 Speicherbelegung

## 5.1 Speicheraufteilung

| Speichertyp | Startadresse | Endadresse | Kommentar |
|---|---|---|---|
| DRAM | $00000000 | $07FFFFFF | 128 MB DRAM |
| Prozessor | $F0000000 | $F00FFFFF | 1 MB Register des MPC5125 |
| ChipRam | $F0200000 | $F0207FFF | 32 KB chiplokales RAM |
| NandFlash | $FFF00000 | $FFFFFFFF | NandFlash Controller Space |

### 5.1.1 DRAM Belegung mit TOP-RTOS (aktuelle System Belegung)

| Speichertyp | Startadresse | Endadresse | Kommentar |
|---|---|---|---|
| PPC-Vector | $00000000 | $00003FFF | 16 KB |
| Anwenderbereich | $00004000 | $075FFFFF | 118 MB - 16 KB |
| **Oberhalb der RTOS-Speicherverwaltung: feste Vergabe der Adressen** | | | |
| DUT | $07600000 | $077FFFFF | 2 MB Display |
| MMU-PTAB | $07800000 | $078FFFFF | 1 MB MMU PageTab |
| .. | $07900000 | $079FEFFF | 1 MB – 4KB . frei |
| PREBOOT | $079FF000 | $079FFFFF | 4KB prebooter |
| RTOS-UH | $07A00000 | $07FFFFFF | Max 6 MB |

### 5.1.2 ChipRam Belegung

| Speichertyp | Startadresse | Endadresse | Kommentar |
|---|---|---|---|
| #FEC | $F0200000 | $F020008F | FEC I/O Bufferpointer |
| FEC-Buffer | $F0200090 | $F0200CFF | 2 FEC Tx-Buffer |
| USB-Verw. | $F0201000 | $F02022FF | USB Verwa. Pointer |
| | $F0202300 | $F0207FFF | #Frei# |

### 5.1.3 NandFlash Belegung

Die Gesamtgröße des NandFlash ist 128 MB (64K Blöcke je 2 KB).

| Start Block | End Block | Size | Bereich |
|---|---|---|---|
| 0 | 255 | 512 KB | frei |
| 256 | 511 | 512 KB | 1.Booter Bereich |
| 512 | 767 | 512 KB | Safe Booter Bereich |
| 768 | 1023 | 512 KB | frei |
| 1024 | 4095 | 6 MB | 1. RTOS Bereich |
| 4096 | 8191 | 8 MB | Safe RTOS-Bereich |
| 8192 | 65535 | 112 MB | Nand-Flash Disk |

### 5.1.4 Vector-Tabelle beim MPC5125

Die Vectortabelle bei einem PowerPC liegt unter RTOS-UH ab der Adresse `$4000`. Die Zuordnung der einzelnen Vectoren für die interne Peripherie beginnt ab `$4120`. Die vollständige Liste ist dem Referenzmanual für den MPC 5125 zu entnehmen.

Hier für die digitale I/O

| Adresse | Hardwarefunktion |
|---|---|
| `$00004124` | GPT10 .. GPT2[2] Kanal |
| `$00004128` | GPT11 ..GPT2[3] Kanal |
| `$0000415C` | GPT0 .. GPT1[0] Kanal |
| `$00004160` | GPT1 .. GPT1[1] Kanal |
| `$000041B8` | GPT8 .. GPT2[0] Kanal |
| `$000041BC` | GPT9 .. GPT2[1] Kanal |
| `$00004240` | GPT2 .. GPT1[2] Kanal |
| `$00004244` | GPT3 .. GPT1[3] Kanal |
| `$00004248` | GPT4 .. GPT1[4] Kanal |
| `$0000424C` | GPT5 .. GPT1[5] Kanal |
| `$00004250` | GPT6 .. GPT1[6] Kanal |
| `$00004254` | GPT7 .. GPT1[7] Kanal |
| `$00004258` | GPIO 0..31 Kanal |
| `$00004278` | GPIO 32..64 Kanal |
| `$00004290` | GPT12 .. GPT2[4] Kanal |
| `$00004294` | GPT13 .. GPT2[5] Kanal |
| `$00004298` | GPT14 .. GPT2[6] Kanal |
| `$0000429C` | GPT15 .. GPT2[7] Kanal |

### 5.1.5 Lokale I/O Geräte auf dem Gerät `/I2C1/`

| Funktion | Bus | Adress | Kommentar |
|---|---|---|---|
| RTC | I2C1 | `$D0` (208) | RealTimeClock |
| EEPROM1 | I2C1 | `$A0` (160) | EEPROM 2KB |
| EEPROM2 | I2C1 | `$A8` (168) | EEPROM 2KB (optional) |
| PWRREG | I2C1 | `$86` (134) | PowerRegister |

## 5.2 RTOS-UH Sumpfzellen

Folgende Sumpfzellen sind mit speziellen Informationen belegt:

| Adresse | Zugriff | Beschreibung |
|---|---|---|
| `$00005300` | Langwort | Taktfrequenz in KHz ($60AE0=396.000) |
| `$00005304` | Langwort | Prozessortyp PPC5125 ($8000 5125) |
| `$00005308` | Langwort | Offset MMU-PTAB zu RAMEND von feste Adresse (TOP) |
| `$000053EA` | Byte | `xxxxxxxB` der Wert wird jede Sekunde aktualisiert (Watch-dog) |
| | | `B` SRAM Batterie OK |
| `$000053EB` | Byte | `xxxRx0xx` Zustand der RTC beim Reset<br>`R` RTC Batterie OK<br>`0` Oszillator Down (beim Reset normal) |

# 6 I/O - Geräte

| Name | LDN | DRV | Beschreibung |
|---|---|---|---|
| /ETH | 17 | 0 | Ethernet-Treiber für 10/100 Mbit |
| /CAN | 123 | 0 | Kanal 1 |
| | | 1 | Kanal 2 |
| /A1 | 0 | 0/2/6 | 1. serielle Schnittstelle RS232 |
| /A2 | 2 | 0/2/6 | 2. serielle Schnittstelle RS232 |
| /A3 | 21 | 0/2/6 | 4. serielle Schnittstelle RS232 (galvanisch getrennt) |
| /I2C | 90 | 0 | Zugriff auf das EEPROM |
| /SPI | 91 | 0..7 | SPI-Interface |
| /H0 | 3 | 0 | uSD-Card |
| /FD | 92 | 0 | NandFlashDisk |
| /AT | 4 | 0/2/6 | LCD-Display Treiber |

## 6.1 I2C

Das 2048 Byte große EEPROM ist über die /I2C/-Dation zu erreichen. Die Adresse, von der gelesen oder geschrieben werden soll, wird im Dateinamen übergeben. Wird keine Adresse übergeben, ist die Startadresse 0. Der Dateiname muß mit einem A beginnen, danach folgt die Adresse 3 stellig:

```
/I2C/C0D160A48      I²C-Bus 1, Gerät 160 ($A0) lesen/schreiben ab Adresse 48
```

Beispiel:

```
handle = OPEN( '/I2C/C0D160A48' ) ;  (* ab Adresse 48   *)
_READ( handle, 20, ADR(buffer) ) ;   (* 20 Bytes lessen *)
_CLOSE( handle ) ;
```

Das EEPROM benötigt bis zu 4 ms zum Schreiben eines Bytes, das Lesen kann bis zu 100 µs dauern.

Weiterhin stehen zum Zugriff auf das EEPROM die folgenden PEARL-Routinen zur Verfügung:

```
I2C_RD_EEPROM( unsigned short adr, unsigned short len, unsigned char *data)
I2C_WR_EEPROM( unsigned short adr, unsigned short len, unsigned char *data)
```

### 6.1.1 Belegung des EEPROM1

Die ersten 48 Byte im EEPROM sind für das System reserviert:

| Adresse | Belegung |
|---|---|
| $0000-$0005 | Physikalische Ethernetadresse (MAC) |
| $0006-$003F | 42 Byte reserviert für das System |
| $0030-$07FF | 2000 Byte frei für Anwenderdaten |

### 6.1.2 Belegung des EEPROM2 (optional)

/I2C/C0D168A0 I²C-Bus 1, Gerät 168 ($A8) lesen/schreiben ab Adresse 0

| Adresse | Belegung |
|---|---|
| $0000-$07FF | 2048 Byte frei für Anwenderdaten |

### 6.1.3 Belegung des PWREG

Alle PINs des PWRREG sind auf Output geschaltet. Das Output-Register ist auf Offset $05 erreichbar.

/I2C/C0D134B5 I²C-BUS 1, Gerät 134 ($86), Offset $05

| BitMask | Ausgang | Bedeutung |
|---|---|---|
| $80 | I2C_IO7 | frei |
| $40 | I2C_IO6 | frei |
| $20 | I2C_IO5 | High : Touch -> Reset ( TRITEC ) |
| $10 | I2C_IO4 | High: Display On ( TRITEC ) |
| $08 | I2C_IO3 | High: DC/DC On/Off ( TRITEC ) |
| $04 | RESET_USB | High: No Reset für USB-PHY |
| $02 | RESET_ETH | High : No Reset für ETH-PHY |
| $01 | PWR_ETH | High: Power für Ethernet Phy |

### 6.1.4 Serielle Schnittstellen

- /A1/ RS232 Bedienschnitstelle (115200 Baud) mit User-Interface
- /A2/ 2. RS232 Schnittstelle (115200 Baud)
- /A3/ 3. RS232 Schnittstelle (115200 Baud)

## 6.2 SPI-Geräte

Der SPI-Treiber unterstützt folgende Geräte auf der RTP1-Basisplatine:

| Name | LDN | DRV | Direction | Datenlänge | Beschreibung |
|---|---|---|---|---|---|
| ADC | 91 | 0 | Write | 2 Byte | Mode für AD-Wandlung einstellen |
| ADC | 91 | 0 | Read | 64 Byte | 16 Werte AI1…AI16 ( 4Byte/Chan) |
| DAC | 91 | 1 | Write | 2 Byte | Alle DA-Kanäle auf diesen Wert setzen |
| DAC | 91 | 1 | Write | 16 Byte | DA-Kanäle 1..8 einzeln setzen |
| PEN | 91 | 2 | Read | 6 Byte | Touch Action Data |
| D1G | 91 | 3 | Read/Write | 4 Byte | DIO17..32 Setzen/Lesen |
| D2G | 91 | 4 | Read/Write | 2 Byte | DO1..16 Setzen/ DI17..31 Lesen |
| DER | 91 | 5 | Read | 1 Byte | DAC-Error auslesen |
| HUP | 91 | 6 | Write | 2 Byte | Wert = 0 : Hupe AUS / sonst AN |

### 6.2.1 ADC Mode Write

| **Bit** | 15 | 14 | 13 | 12 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 24 | Conti | 5V | unipolar | | | | Sampletime in µs | | | | | | | |
| 0 | 16 | Single | 10V | bipolar | | | | | | | | | | | |

### 6.2.2 ADC Read

Das Anwenderprogramm muss einen Buffer von insgesamt 64 Byte (16 Werte á 32 Bit) zur Verfügung stellen, in die beim Read die aktuellen Werte der 16 AD-Kanäle eingefüllt werden. Die Werte sind folgendermaßen aufgebaut:

| Bit | Bedeutung |
|---|---|
| `0x01000000` | Overflow-bit |
| `0x02000000` | Sign–Bit im Bipolar Betrieb bei Overlow |
| `0x00`*xxxxxx* | 24 Bit AD-Wert bei 24 Bit-Wandlung |
| `0x0000`*xxxx* | 16 Bit AD-Wert bei 16 Bit Wandlung |

Beispiel: 24 Bit Auflösung / ± 10 Volt Bereich:

```
DCL adc(16)  BIT(32) ;
DCL volt(16) FLOAT   ;
DCL bitval   FLOAT   ;
DCL offset   BIT(32) INIT('00800000'B4 ) ;
DCL mask     BIT(32) INIT('00FFFFFF'B4 ) ;

Bitval = 10.0 / ( 32767.0 * 256.0 ) ;
…
READ adc FROM ADC ; /* einlesen der AD-Rohwerte */
FOR i TO 16 REPEAT;
   volt(i) =  bitval * TOFIXED(( adc(i) AND mask ) - offset );
END ;
```

### 6.2.3 DAC Write

Das Anwenderprogramm muss einen Buffer von 8 Werten á 16 Bit zur Verfügung stellen, aus diesen wird beim Write dann in die 8 Kanäle des DA-Wandlers geschrieben. Es werden nur die unteren 12 Bit genutzt, da ein 12 Bit-DA-Wandler verbaut ist.

### 6.2.4 PEN Read

Von dem Geräte kann eine 6 Byte grosse Struktur gelesen werden, die folgenden Inhalt enthält:

| | |
|---|---|
| Action.W | (1=Touch erkannt, 2=Schieben, 0=kein Touch) |
| PosX.W | aktuelle Displaykoordinate X |
| PosY.W | aktuelle Displaykoordinate y |

Das System löst einen EV 00000001 aus beim UP/DOWN.

Das System löst einen EV 00000002 aus beim Schieben

### 6.2.5 D1G Read / Write

Es werden die digitalen Eingänge 33..64 besetzt bzw. eingelesen.

### 6.2.6 D2G Read / Write

Beim Read werden die digitalen Eingänge 17..32 eingelesen. Beim Write werden die digitalen Ausgänge 1..16 gesetzt.

### 6.2.7 DER Read

Es wird der Status der DAC-Kanäle gelesen. Damit kann der DAC überwacht werden.

### 6.2.8 HUP Write

Es ist ein Wort zu übergeben. Mit dem Wert `0` wird der Signalgeber ausgeschaltet, jeder andere Wert schaltet ihn ein.

## 6.3 NandFlashDisk /FD/

Die resetfeste NandFlashDisk /FD/ liegt im lokalen Nand-Flash, sie ist maximal 110 MB groß.

Der Formatbefehl lautet: FORM D /FD/FULL

## 6.4 Autostart

Ist auf der µSD-Card /H0/ eine Datei `AUTO.EX` vorhanden, so wird diese beim Programmstart mit der EX-Shell abgearbeitet, d.h. alle Zeilen dieser Datei werden der Reihe nach als Befehl interpretiert und ausgeführt. Ist keine µSD-Karte eingelegt oder keine Datei `AUTO.EX` vorhanden, wird auf der NandFlashDisk `/FD/` weiter gesucht. Wird dort die Datei `AUTO.EX` gefunden, wird sie entsprechend abgearbeitet.

Startet das Recover RTOS, wird zuerst auf der µSD-Card /H0/ nach der Datei `RECOVER.EX` gesucht. Ist sie vorhanden, wird sie mit der EX-Shell abgearbeitet, d.h. alle Zeilen dieser Datei werden der Reihe nach als Befehl interpretiert und ausgeführt. Ist keine µSD-Karte eingelegt oder keine Datei `RECOVER.EX` gefunden worden, wird auf der NandFlashDisk `/FD/` weiter gesucht. Wird dort die Datei `RECOVER.EX` gefunden, wird sie entsprechend abgearbeitet.

## 6.5 NotStart / Recover RTOS

Das System überprüft den Booter bzw. das RTOS-UH im NAND-Flash beim Starten mit einer Checksumme. Wenn der 1. Booter OK ist und das 1. RTOS-UH - die normalen Arbeitsversionen - fehlerfrei sind (Checksumme stimmt), dann wird dieses RTOS-UH gestartet und der Autostart mit der `AUTO.EX` durchgeführt. Falls die Checksumme nicht stimmt, dann wird das Safe-RTOS gestartet und der Autostart mit der `RECOVER.EX` durchgeführt.

# 7 RTOS-UH im FLASH ablegen

Nachdem das RTOS-UH als S-Record auf die Adresse `$800000` geladen wurde und der Befehl `START_RTOS` ausgeführt wurde, kann das dann laufende neue RTOS mit dem Befehl

| | |
|---|---|
| `NAND S` | System wird im Nand-Flash abgelegt. |
| `NAND B` | schreibt den Booter in das Nand-Flash |
| `NAND F` | schreibt den SafeBooter und das SafeRTOS in das Nand-Flash. |

# 8 Terminalemulation

## 8.1 Allgemeines

Die Terminalemulation ermöglicht den Anschluß Displays an einem RTOS-UH-Rechner. Unter den mnemotechnischen Bezeichnungen `/AT`, `/BT` und `/CT` ist die Terminalemulation in der A-, B- oder C-Betriebsart unter der LDN 4 mit den Drives 0, 2 und 6 erreichbar. Unter der Bezeichnung `/DT` wird der Ausgabekanal im Vollduplex-Betrieb mit der LDN 14 angesprochen.

## 8.2 Befehlsvorrat Terminalemulation

Die Terminalemulation verhält sich weitgehend Televideo-kompatibel. Sie versteht die folgende Befehlssequenzen (Erläuterung der Angaben *Ps*, *Pc*, *Pn*, *r* und *c* siehe unten). Es ist nicht gewährleistet, daß alle Funktionen auf allen Terminals zur Verfügung stehen! So können auf einen schwarz/weiß-Display natürlich keine Farben eingestellt werden, oder auf einen Textdisplay wird es schwierig, Grafik auszugeben.

Beachten Sie bitte, dass die folgenden Sequenzen nur nutzbar sind, wenn der Grafik-Mode nicht benutzt wird!

| Cursor style (0..4) | | | |
|---|---|---|---|
| | ESC . *Ps* | | |
| | | 0 | Cursor off |
| | | 1 | Blinking block |
| | | 2 | Steady block |
| | | 3 | Blinking underline |
| | | 4 | Steady underline |

| Define Visual attributes | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ESC G *Ps* | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| | | Underline | | | | | | | | | X | X | X | X | X | X | X | X |
| | | Reverse | | | | | X | X | X | X | | | | | X | X | X | X |
| | | Blinking | | | X | X | | | X | X | | | X | X | | | X | X |
| | | Invisible | | X | | X | | X | | X | | X | | X | | X | | X |

| Operating Modes | | | |
|---|---|---|---|
| | ESC | ) | Enable invers |
| | | ( | Disable invers |
| | | V | Autoscroll off |
| | | W | Autoscroll on |
| | | $ | Grafics mode on |
| | | % | Grafics mode off |
| | | U | Monitor mode on |
| | | X | Monitor mode off |
| | | [=7h | Auto Wrap on |
| | | [=7l | Auto Wrap off |
| | | ! *c* | c >= ‚A‘: Schreibfarbe, c<‘A‘: Hintergrundfarbe setzen |
| | | [ *Pc* ; *Pc* r | Set scroll region rows (*Pc*: start, end) |
| | | [ *Pc* ; *Pc* s | Set scroll region columns (*Pc*: start, end) |

| Cursor Movement | | | |
|---|---|---|---|
| | CTRL | J | Line feed |
| | | K | Up |
| | | V | Down |
| | | L | Right |
| | | M | Carriage Return |
| | | H | Left |
| | ESC | = *r c* | Set Cursor *row column* |
| | | [ *Pn* A | Cursor up *Pn* rows |
| | | [ *Pn* B | Cursor down *Pn* rows |
| | | [ *Pn* C | Cursor right *Pn* columns |
| | | [ *Pn* D | Cursor left *Pn* columns |
| | | [ *c* ; *r* H | Set Cursor position *column*, *row* |

| Tabulator Control | | | |
|---|---|---|---|
| | ESC | 1 | Set tab stop at actual column |
| | | 2 | Clear tab stop at actual column |
| | | 3 | Clear all tab stops |
| | | i | Move Cursor to next tab stop |
| | | I | Move Cursor back one tab stop |

| Insertion | | | |
|---|---|---|---|
| | ESC | Q | Insert space at cursor |
| | | E | Insert line of spaces |

| Miscellaneous | | | |
|---|---|---|---|
| | ESC | ~ | Reset Terminal |

| Deletion | | | |
|---|---|---|---|
| | | W | Delete char at cursor |
| | | R | Delete current line to spaces |
| | | T | Delete to end of line |

| | | | |
|---|---|---|---|
| | | t | |
| | | Y | Delete to end of screen |
| | | y | |
| | | * | Delete Screen |
| | | , | |
| | | ; | |
| | | + | |
| | | : | |
| | CTRL Z | | |

Die Befehlssequenz zur Änderung der Schreibfarbe (`ESC ! c`) ist nicht zu gängigen Terminalemulationen kompatibel. Neben der einfachen Änderung der Schreibfarbe kann durch Änderung des Grundwertes der Farbangabe folgendes Verhalten erzeugt werden:

| Grundwert | Verhalten |
|---|---|
| 32 (Leerzeichen) | Änderung der Schreibfarbe |
| 64 (A) | Änderung der Hintergrundfarbe |

Die Angaben für mit *Ps*, *Pc*, *Pn*, *r* oder *c* gekennzeichnete Zahlenwerte erfolgen durch Werte, die sich ASCII-codiert aus 32 + Zahlenwert errechnen.

## 8.3 Grafikfunktionen

Bei grafikfähigen Displays stellt die Terminalemulation Funktionen zur Nutzung der grafischen Fähigkeiten des Displays zur Verfügung. Der Leistungsumfang richtet sich nach der Leistungsfähigkeit des jeweiligen Grafik-Displays bzw. des verwendeten Controllers.

Die Grafikfunktionen sind aus Geschwindigkeitsgründen nicht auf Multi-Tasking ausgelegt. Grundsätzlich sollte ein Grafiksystem nur von einer Task angesprochen werden. Besonders der kombinierte Betrieb von Terminalemulation und Grafikfunktionen erfordert besondere Aufmerksamkeit.

Bitte beachten Sie, dass bei der Nutzung des Grafik-Mode bei bestimmten Displays der Attributspeicher nicht mehr zur Verfügung steht. Dies hat z.B. zur Folge, dass die Ausgabe von invertiertem Text nicht mehr möglich ist.

Neben den Grafik-Primitiven `SETPIX`, `GETPIX` und `LINE` (s. RTOS-UH-Handbuch Kapitel 5.7.3) gibt es folgende Prozeduren:

### 8.3.1 Funktionsumfang und Implementierungsabhängigkeiten

| Prozedur | IP-SVGA | PVGA | VHI | NBS300 | MPC555 |
|---|---|---|---|---|---|
| BACK_PEN | Ja | Ja | — | — | — |
| BIN_TEXT | Ja | Ja | Ja | Ja | Ja |
| BOX | Ja | Ja | Ja | Ja | Ja |
| BOX_FILLED | Ja | Ja | Ja | Ja | Ja |
| BOX_MOVE | Ja | Ja | Ja | Ja | Ja |
| BOX_READ | Ja | Ja | — | — | — |
| BOX_WRITE | Ja | Ja | — | — | — |
| CIRCLE | Ja | Ja | Ja | Ja | Ja |
| CLEAR | Ja | Ja | Ja | Ja | Ja |
| COLOR_CLEAR | Ja | Ja | — | — | — |
| DISPLAY_MODE | Ja | Ja | Ja | Ja | Ja |
| DISPLAY_STATE | Ja | Ja | — | — | — |
| DOT_LINE | Ja | Ja | Ja | Ja | Ja |
| DOT_LINE_RANGE | Ja | Ja | Ja | Ja | Ja |
| FILLED_HISTOGRAM | Ja | Ja | Ja | Ja | Ja |
| GET_PEN | Ja | Ja | — | — | — |
| GET_BACK_PEN | Ja | Ja | — | — | — |
| GET_TEXT_FONT | Ja | Ja | — | — | — |
| GET_TEXT_WIDTH | Ja | Ja | Ja | Ja | Ja |
| GRAPH_CURSOR | Ja | Ja | — | — | — |
| GRAPH_CURSOR_SELECT | Ja | Ja | — | — | — |
| HIDE_CURSOR | Ja | Ja | — | — | — |
| HISTOGRAM | Ja | Ja | Ja | Ja | Ja |
| PEN | Ja | Ja | — | — | — |
| PLANE | — | Ja | — | — | — |
| PLINIT | Ja | Ja | Ja | Ja | Ja |
| POLYGON | Ja | Ja | Ja | Ja | Ja |
| SELECT_FONT | Ja | Ja | — | — | — |
| SELECT_TEXT_FONT | Ja | Ja | — | — | — |
| SET_TEXT_WIDTH | Ja | Ja | Ja | Ja | Ja |
| SET_VGA_SCREEN | Ja | Ja | — | — | — |
| TEXT | Ja | Ja | Ja | Ja | Ja |
| VGA_GET_PLANE | Ja | Ja | — | — | — |
| VGA_LOCK_TEXT | Ja | Ja | — | — | — |
| VGA_SET_PLANE | Ja | Ja | — | — | — |
| VGA_UNLOCK_TEXT | Ja | Ja | — | — | — |
| WIDTH | Ja | Ja | Ja | Ja | Ja |

### 8.3.2 Allgemeines

Zum Verständnis der Funktionsbeschreibung sind Kenntnisse über die folgenden Begriffe erforderlich:

- Farbe

  Farben werden in RGB-Notation angegeben. Die Angabe kennzeichnet die Intensität, in der der jeweilige Farbanteil dargestellt wird.

- Pixel

  Kennzeichnet einen Bildpunkt. Je nach verwendetem Grafiksystem kann ein Bildpunkt unterschiedliche Farben annehmen.

- Pen

  Ein Pen ist durch eine Farbe gekennzeichnet.

  Es gibt einen Schreib- und einen Hintergrund-Pen. Pixel, Linien u.ä. werden nur unter Verwendung des Schreib-Pens gezeichnet.

  Vorgegebene Konstrukte wie Fonts o.ä. füllen den Zeichenhintergrund mit dem Hintergrund-Pen und stellen die Zeichen mit dem Schreib-Pen dar.

- Font

  Ein Font (Zeichensatz) ist als Bitmap abgelegt. Alle Zeichen eines Fonts haben gleiche Größe. Die Größe eines Fonts beinhaltet Ober- und Unterlängen.

  Die Bitmap enthält Informationen darüber, welche Pixel der von einem Zeichen eingenommen Fläche mit dem Schreib-Pen und welche Pixel mit dem Hintergrund-Pen darzustellen sind. Ein Font selber enthält keine Farbinformationen.

- Palette oder Plane

  Die im Bildschirmspeicher abgelegten Pixel-Farbinformationen werden über Paletten in tatsächliche Farbintensitätswerte übersetzt.

  Die Pixel-Farbinformation kann in diesem Fall mit geringem Speicherbedarf abgelegt werden (z.B. 4 Bit). Diese Information wird als Index beim Zugriff auf die Palette verwendet. Die Palette liefert dann die tatsächliche Farbinformation als RGB-Wert mit deutlich höherer Farbauflösung (z.B. 3x 8 Bit).

- Screen

  Ein Screen ist die Menge aller für einen sichtbaren Bildschirm im Bildspeicher abgelegten Daten. Bei Grafiksystemen, die entweder mehrere unterschiedliche Bildschirme unterstützen oder im Bildspeicher gleichzeitig Daten für mehrere Bildschirme ablegen können, ist die Wahl des Screens, auf dem die folgenden Zeichenoperationen erfolgen sollen, möglich.

### 8.3.3 Funktionsreferenz

| **PEARL-Spezifikation** | **Funktion** |
|---|---|
| **BACK_PEN:** PROC(<br>col FIXED(15)<br>) GLOBAL; | Setzt die angegebene Farbe `col` als Hintergrundfarbe. Bei folgenden Operationen, die eine Hintergrundfarbe darstellen müssen, aber keine eigene Hintergrundfarbe spezifizieren ( `TEXT` und `BIN_TEXT`), wird `col` als Hintergrundfarbe verwendet. |
| **BIN_TEXT:** PROC(<br>(x, y) FIXED(15), | Überträgt ein Zeichen aus dem Zeichengenerator-ROM in der Farbe `col` auf den Bildschirm an die Position (`x`, |

| | |
|---|---|
| col FIXED(15),<br>zeichen_adr FIXED(31)<br>) GLOBAL; | y). Die Adresse im ROM wird mit zeichen_adr angegeben. Sie hängt von der Größe des auszugebenden Zeichens ab:<br>$zeichen\_adr = \frac{xhöhe * ybreite + 7}{8}$ |
| **BOX:** PROC(<br>(x, y) FIXED(15),<br>(Xend, Yend ) FIXED(15),<br>col FIXED(15)<br>) GLOBAL; | Zeichnet einen rechteckigen Rahmen in der Farbe col mit den diagonalen Punkten (x, y) (linke obere Ecke) und (Xend, Yend) (rechte, untere Ecke). |
| **BOX_FILLED:** PROC(<br>(x, y) FIXED(15),<br>(Xend, Yend ) FIXED(15),<br>col FIXED(15)<br>) GLOBAL; | Zeichnet ein in der Farbe col gefülltes Rechteck mit den diagonalen Punkten (x, y) und (Xend, Yend) |
| **BOX_MOVE:** PROC(<br>(Xstart, Ystart) FIXED(15),<br>(Breite, Hoehe ) FIXED(15),<br>(Xziel, Yziel ) FIXED(15),<br>mode FIXED(15)<br>) GLOBAL; | Kopiert einen rechteckigen Bildausschnitt von Breite und Höhe beginnend bei dem Startpunkt (Xstart, Y-start) auf einen Bildbereich gleicher Größe mit dem Startpunkt (Xziel, Yziel).<br>Der Zielbereich kann außerhalb des sichtbaren Bildschirmbereichs liegen, d.h. Ywidth wird überschritten.<br>mode legt die Zeichenart fest, im unteren Nibble sind folgende Werte zulässig:<br><br>**Dez. hex Art**<br>12 $0C absolut<br>3 $03 not<br>8 $08 and<br>14 $0E or<br>6 $06 exor<br><br>Das nächste Nibble legt die Schreibfarbe fest, d.h. für ein Pixel werden die 4 Bit aus dem Video-RAM entsprechend dem unteren Nibble gelesen, mit der Schreibfarbe „verundet“ und wieder im Video-RAM abgelegt. Wird der Parameter mode auf absolut (=12) gesetzt, so wird der original Bildausschnitt an der Zielstelle abgelegt. |
| **BOX_READ:** PROC(<br>(Xstart, Ystart) FIXED(15),<br>(Breite, Hoehe ) FIXED(15),<br>feld STRUCT[ REF CHAR(1) ]<br>) GLOBAL; | Kopiert einen rechteckigen Bildausschnitt mit Breite und Hoehe beginnend bei dem Startpunkt (Xstart, Ystart) in den Speicherbereich, der mit feld angegeben wird. Dabei findet **keine** Überprüfung der Größe des Speicherbereiches statt.<br>Ist das angegebene feld zu klein, ist der Absturz des Rechners so gut wie sicher! |
| **BOX_WRITE:** PROC(<br>(Xstart, Ystart) FIXED(15),<br>(Breite, Hoehe ) FIXED(15),<br>mode FIXED(15),<br>feld STRUCT[ REF CHAR(1) ]<br>) GLOBAL; | Kopiert den Speicherbereich, der mit feld angegeben wird, auf einen rechteckigen Bildausschnitt von Breite und Höhe, beginnend bei dem Startpunkt (Xstart, Y-start).<br>Dabei findet **keine** Überprüfung der Größe des Speicherbereiches statt. Ist das angegebene feld zu klein, |

| | |
|---|---|
| | wird der folgende Speicher auf den Bildschirm geschrieben.<br>Der Parameter `mode` ist bei der Prozedur `BOX_MOVE` beschrieben. |
| **CIRCLE:** PROC(<br>(Xmitte, Ymitte) FIXED(15),<br>Radius FIXED(15),<br>col FIXED(15)<br>) GLOBAL; | Zeichnet geschlossene Kreislinien auf den Bildschirm. |
| **CLEAR:** PROC GLOBAL; | Löscht den sichtbaren Bildschirmbereich mit der Farbe 0. |
| **COLOR_CLEAR:** PROC(<br>col FIXED(15)<br>) GLOBAL; | Löscht den sichtbaren Bildschirmbereich mit der angegebenen Farbe `col`. |
| **DISPLAY_MODE:** PROC(<br>mode FIXED(31)<br>) GLOBAL; | Ermöglicht bei einigen Systemen die Umschaltung der Betriebsart des Grafiksystems (Text=0/Grafik=1). |
| **DISPLAY_STATE:** PROC(<br>State FIXED(31)<br>) GLOBAL; | Schaltet Display ein (`State` = 1) oder aus (`State` = 0) |
| **DOT_LINE:** PROC(<br>(Xstart, Ystart) FIXED(15),<br>(XEnd, YEnd) FIXED(15),<br>col FIXED(15),<br>mask FIXED(15)<br>) GLOBAL; | Zeichnet ein (strichpunktierte) Linie vom Punkt (`Xstart`, `Ystart`) zum Punkt (`XEnd, YEnd`) in der Farbe `col`.<br>Mit `mask` wird das Linienmuster vorgegeben:<br>-1: durchgezogene Linie<br>TOFIXED ´AAAA'B4: einfach punktiert |
| **DOT_LINE_RANGE:** PROC(<br>(Xstart, Ystart) FIXED(15),<br>(XEnd, YEnd) FIXED(15),<br>col FIXED(15),<br>mask FIXED(15),<br>(YU, YO) FIXED(15),<br>col2 FIXED(15),<br>) GLOBAL; | Zeichnet ein (strichpunktierte) Linie vom Punkt (`Xstart`, `Ystart`) zum Punkt (`XEnd, YEnd`) in der Farbe `col`.<br>Mit `mask` wird das Linienmuster vorgegeben:<br>-1: durchgezogene Linie<br>TOFIXED ´AAAA'B4: einfach punktiert<br>Die Y-Koordinaten `YU` (unten) und `YO` (oben) sind Umschaltschwellen für die Zeichenfarbe. Liegt ein zu zeichnender Bildpunkt unterhalb `YU` oder oberhalb `YO`, so wird er in der Farbe `col2` anstatt in `col` dargestellt. |
| **FILLED_HISTOGRAMM:** PROC(<br>(XS, YS) FIXED(15) IDENT,<br>count FIXED(15),<br>col FIXED(15),<br>b_col FIXED(15),<br>width FIXED(15),<br>ySave FIXED(15) IDENT<br>) GLOBAL; | Zeichnet ein Histogramm aus `count` nicht gefüllten Balken der Breite `width` in der Farbe `col`.<br>Die einzelnen Histogrammbalken werden durch ihre linke, obere Ecke (`XS, YS`) und die Breite `width` bestimmt. `XS` und `YS` müssen in getrennten FIXED(15)-Feldern gegeben werden.<br>Die Y-Grundlinie des Histogramms wird durch die Initialwerte im Feld `ySave` gegeben.<br>Zur Erhöhung der Zeichengeschwindigkeit bei wiederholter Darstellung eines Histogramms wird nur die Änderung der Balkenhöhe gegenüber der im Feld `ySave` gespeicherten Höhe neu gezeichnet. |
| **GET_PEN:** PROC<br>RETURNS( FIXED(15) )<br>GLOBAL; | Gibt die Farbe des aktuell gewählten Schreib-Pens zurück. |
| **GET_BACK_PEN:** PROC<br>RETURNS( FIXED(15) ) | Gibt die Farbe des aktuell gewählten Hintergrund-Pens zurück. |

| | |
|---|---|
| GLOBAL; | |
| **GET_TEXT_FONT:** PROC<br>  RETURNS( FIXED(15) )<br>GLOBAL; | Gibt die Nummer des aktuellen Textfonts zurück (von SELECT_FONT oder SELECT_TEXT_FONT gesetzt). Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem. |
| **GET_TEXT_WIDTH:** PROC(<br>  Xhoehe  FIXED(15) IDENT,<br>  ybreite  FIXED(15) IDENT<br>) GLOBAL; | Liefert die Größe eines Zeichens des gewählten Zeichensatzes in den Variablen xhoehe und ybreite zurück. |
| **GRAPH_CURSOR:** PROC(<br>  (x, y)  FIXED(15),<br>  status  FIXED(15)<br>) GLOBAL; | Positioniert den Graphik-Cursor auf den Punkt (x, y). Mit status = 1 wird der Graphik-Cursor sichtbar, mit status = 0 wird er unsichtbar. |
| **GRAPH_CURSOR_SELECT:** PROC(<br>  Cursor  FIXED(31) IDENT,<br>  mode  FIXED(15)<br>) GLOBAL; | Gibt eine Bitmap zur Darstellungsform des Graphik-Cursors vor. Cursor muß das erste Element eines FIXED(31)-Feldes mit 32 Elementen sein; damit wird ein 32x32 Bit-Cursor beschrieben. Gesetzte Bits entsprechen darzustellenden Pixeln.<br>mode gibt die Darstellungsart des Cursors an. |
| **HIDE_CURSOR:** PROC GLOBAL; | Schaltet den Graphik-Cursor aus. |
| **HISTOGRAMM:** PROC(<br>  (XS, YS)  FIXED(15) IDENT,<br>  count  FIXED(15),<br>  col  FIXED(15),<br>  b_col  FIXED(15),<br>  (offs, width)  FIXED(15),<br>  ySave  FIXED(15) IDENT<br>) GLOBAL; | Zeichnet ein Histogramm aus count nicht gefüllten Balken der Breite width in der Farbe col.<br>Die Y-Grundlinie des Histogramms wird in offs gegeben.<br>Die einzelnen Histogrammbalken werden durch ihre linke, obere Ecke (XS, YS) und die Breite width bestimmt.<br>XS und YS müssen in getrennten FIXED(15)-Feldern gegeben werden.<br>Zur Erhöhung der Zeichengeschwindigkeit bei wiederholter Darstellung eines Histogramms wird nur die Änderung der Balkenhöhe gegenüber der im Feld ySave gespeicherten Höhe neu gezeichnet. Bei der erstmaligen Darstellung eines Histogramms muss ySave daher vollständig mit dem Wert offs initialisiert werden. |
| **PEN:** PROC(<br>  col  FIXED(15)<br>) GLOBAL; | Setzt die angegebene Farbe col als Schreibfarbe. Bei folgenden Operationen, die keine eigene HinSchreibfarbe spezifizieren, wird col als Schreibfarbe verwendet. |
| **PLANE:** PROC(<br>  Farbnummer  FIXED(15),<br>  Palettenwert FIXED(31)<br>) GLOBAL; | Setzt den Wert eines Paletteneintrags. Für Farbnummer sind Werte von 0 bis 15 zulässig. Folgende Bits vom Palettenwert werden genutzt (abhängig vom Grafiksystem):<br><br>**Farbe  Bit (PEARL)  Bit(Assembler)**<br>Rot  12 ... 4  18 ... 20<br>Grün  20 ...22  10 ... 12<br>Blau  28 ...30  2 ... 4<br><br>Der Aufbau des Langwortes Palettenwert läßt sich durch die Einzelbit-Darstellung |

| | |
|---|---|
| | [---- ---- ---r rr-- ---g gg-- ---b bb--] veranschaulichen. |
| **PLINIT:** PROC GLOBAL; | Initialisiert den Display-Prozessor. Muß vor Benutzung der Grafik-Routinen einmal aufgerufen werden. |
| **POLYGON:** PROC(<br>(x,y) FIXED(15) IDENT,<br>cnt FIXED(15,<br>col FIXED(15)<br>) GLOBAL; | Zeichnet einen (nicht geschlossenen) Linienzug mit `cnt` Stützpunkten in der Farbe `col`.<br>`X` und `y` müssen in getrennten, eindimensionalen Feldern abgelegt sein. |
| **SELECT_FONT:** PROC(<br>Font FIXED(15)<br>) GLOBAL; | Wählt den angegeben Font als aktuellen Textfont aus. Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem. |
| **SELECT_TEXT_FONT:** PROC(<br>Font FIXED(15)<br>) GLOBAL; | Wählt den angegeben Font als aktuellen Textfont aus. Anzahl und Art der verfügbaren Fonts sind abhängig vom eingesetzten Grafiksystem. |
| **SET_TEXT_WIDTH:** PROC(<br>Xhöhe FIXED(15) IDENT,<br>ybreite FIXED(15) IDENT<br>) GLOBAL; | Setzt die Größe eines Zeichens für die Funktionen `BIN_TEXT` und `TEXT`. |
| **SET_VGA_SCREEN:** PROC(<br>Screen FIXED(15)<br>)<br>RETURNS( FIXED(15) )<br>GLOBAL; | Wählt den angegebenen `Screen` als Ziel der folgenden Graphikoperationen aus.<br>Der Rückgabewert ist –1, falls der angegebene `Screen` im aktuellen Grafiksystem nicht vorhanden ist. |
| **TEXT:** PROC(<br>(x, y) FIXED(15),<br>col FIXED(15),<br>string STRUCT<br>[ str CHAR(255),<br>len FIXED(15)<br>]<br>) GLOBAL; | Schreibt die in `string.str` abgelegte Zeichenkette beginnend auf der Position (`x`, `y`) (linke, obere Ecke des ersten Zeichens) in der Farbe `col` auf den Bildschirm.<br>Die Ausgabe endet, wenn `string.len` Zeichen ausgegeben wurden.<br>Die Schreibrichtung ist horizontal (x-Richtung). |
| **VGA_GET_PLANE:** PROC(<br>Plane FIXED(15),<br>(R, G, B) FIXED(15) IDENT<br>) GLOBAL; | Liest aus der Palette `Plane` den enthaltenen RGB-Wert aus. |
| **VGA_LOCK_TEXT:** PROC<br>RETURNS( FIXED(15) )<br>GLOBAL; | Sperrt das Grafiksystem gegenüber den Aktionen der Terminalemulation.<br>Diese Funktion muß jeweils vor einem Block von zusammenhängenden Grafikoperationen aufgerufen werden, falls gleichzeitig die Terminalemulation genutzt werden soll.<br>Der Rückgabewert ist 1, falls die Operation erfolgreich war. |
| **VGA_SET_PLANE:** PROC(<br>Plane FIXED(15),<br>(R, G, B) FIXED(15)<br>) GLOBAL; | Setzt die Palette `Plane` auf einen angegebenen RGB-Wert. Plane wird beim Zeichnen über die `col` angesprochen |
| **VGA_UNLOCK_TEXT:** PROC<br>RETURNS( FIXED(15) )<br>GLOBAL; | Gibt das Grafiksystem nach einem Aufruf von `VGA_LOCK_TEXT` wieder für Aktivitäten der Terminalemulation frei.<br>Der Rückgabewert ist 1, falls die Operation erfolgreich war. |

| | |
|---|---|
| **WIDTH:** PROC(<br>  Xwidth  FIXED(15) IDENT,<br>  Ywidth  FIXED(15) IDENT<br>) GLOBAL; | Gibt in den Variablen `Xwidth` und `Ywidth` die Größe des Bildschirms in Pixel zurück. |

### 8.3.4 Zeichensatzgrößen

Die Terminalemulation stellt verschieden große Zeichensätze zur Verfügung.

| Zeichensatz | Zeichengröße in Pixel |
|---|---|
| 0 | 8x16 |
| 1 | 8x8 |
| 2 | 4x6 |
| 3 | 6x11 |
| 4 | 7x14 |
| 5 | 10x18 |
| 6 | 12x22 |
| 7 | 4x6 |
| 8 | 5x8 |
| 9 | 5x12 |
| 10 | 6x8 |
| 11 | 6x10 |
| 12 | 7x12 |
| 13 | 7x12 B |
| 14 | 8x8 |
| 15 | 8x12 |
| 16 | 8x14 |
| 17 | 10x16 |
| 18 | 12x16 |
| 19 | 12x20 |
| 20 | 16x26 |
| 21 | 22x36 |
| 22 | 24x40 |
| 23 | 32x53 |

### 8.3.5 Zeichensatz einstellen

Mit der Funktion

***SELECT_TEXT_FONT( FontNr FIXED(15) )***

wird die Zeichensatzgröße eingestellt.

Standardmäßig ist der FontNr = 0 ( 8x16 Pixel ) eingestellt.

Falls größere Zeichen benötig werden besteht die Möglichkeit durch die Angabe von FontNr + 256 die Zeichen in doppelter Breite/Höhe zu zeichnen.

Falls FontNr+512 angegeben wird, dann wird die ANSI-Zeichentabelle anstatt der DOS-Tabelle für die Darstellung genutzt. D.h Zeichen oberhalb von 0x80 werden anders gemapped.

### 8.3.6 Zeichensatz auslesen

Der aktuell eingestellte Zeichensatz, kann mit der Funktion

***GET_TEXT_FONT RETURNS( FIXED(15) )***

ausgelesen werden.

Die Zeichengröße eines Zeichens kann mit der Funktion

***GET_TEXT_WIDTH( b FIXED(15) IDENT , h FIXED(15) IDENT )***

abgefragt warden.

### 8.3.7 Zeichensatz Farben

Die zum Zeichen der Zeichen benutze Farbe kann mit der Funktion

***PEN( fcolor FIXED(15) )***

eingestellt werden. Standard ist ***WHITE.***

Für den Zeichenhintergrund wird die Farbe genutzt die mit der Funktion

***BACK_PEN( bcolor FIXED(15) )***

eingestellt ist. Standard ist ***BLACK.***

### 8.3.8 Farbdarstellung

Die Grafik / Zeichen werden im BGR Format mit 5x5x5 Bit Auflösung dargestellt, daher ergeben sich folgende Farbmasken für den Wert einer Farbe

| Farbanteil | Farbmaske |
|---|---|
| RED | 0x001F |
| GREEN | 0x03E0 |
| BLUE | 0x7C00 |

Daher setzt sich der Farbwert folgende zusammen:
b,g,r sind die Farbstärke im Bereich von (dunkel) 0..31 (hell)

```
color = TOFIXED(  (( TOBIT( b ) SHIFT 10 ) AND ’7C00’B4 )
                  OR (( TOBIT( g ) SHIFT 5 ) AND ’03E0’B4 )
                  OR ( TOBIT( r ) AND ’001F’B4 )  )  ;
```

### 8.3.9 Zeichensatz

Der Zeichensatz für die Funktion TEXT enthält folgende Zeichen oder den eingebauten Zeichensatz des Displays:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ☺ | ☻ | ♥ | ♦ | ♣ | ♠ | ● | ◘ | ○ | ◙ | ♂ | ♀ | ♪ | ♫ | * |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| ► | ◄ | ↕ | ‼ | ¶ | § | ■ | ↨ | ↑ | ↓ | → | ← | ∟ | ↔ | ▲ | ▼ |
| 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| | ! | ” | # | $ | % | & | ’ | ( | ) | * | + | , | - | . | / |
| 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 |
| @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 80 | 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 |
| P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 96 | 97 | 98 | 99 | 100 | 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 | 111 |
| ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 | 121 | 122 | 123 | 124 | 125 | 126 | 127 |
| p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | ⌂ |
| 128 | 129 | 130 | 131 | 132 | 133 | 134 | 135 | 136 | 137 | 138 | 139 | 140 | 141 | 142 | 143 |
| | ű | é | ξ | ä | à | á | ς | ê | ë | è | ï | î | ì | Ä | Ã |
| 144 | 145 | 146 | 147 | 148 | 149 | 150 | 151 | 152 | 153 | 154 | 155 | 156 | 157 | 158 | 159 |
| É | æ | Æ | ô | ö | ò | û | ù | Ÿ | Ő | Ű | ¢ | £ | ¥ | Pts | *f* |
| 160 | 161 | 162 | 163 | 164 | 165 | 166 | 167 | 168 | 169 | 170 | 171 | 172 | 173 | 174 | 175 |
| á | é | í | ú | ń | Ń | | | ¿ | ⌐ | ¬ | ½ | ¼ | i | « | » |
| 176 | 177 | 178 | 179 | 180 | 181 | 182 | 183 | 184 | 185 | 186 | 187 | 188 | 189 | 190 | 191 |
| ▒ | ▒ | ▓ | ▏ | ┤ | ╡ | ╢ | ╖ | ╕ | ╣ | ║ | ╗ | ╝ | ╜ | ┐ | |
| 192 | 193 | 194 | 195 | 196 | 197 | 198 | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 |
| └ | ┴ | ┬ | ├ | ─ | ┼ | ╞ | ╟ | ╚ | ╔ | ╩ | ╦ | ╠ | ═ | ╬ | ╧ |
| 208 | 209 | 210 | 211 | 212 | 213 | 214 | 215 | 216 | 217 | 218 | 219 | 220 | 221 | 222 | 223 |
| ╨ | ╤ | ╥ | ╙ | ╘ | ╒ | ╓ | ╫ | ╪ | ┘ | ┌ | ▉ | ▄ | ▌ | ▐ | ▀ |
| 224 | 225 | 226 | 227 | 228 | 229 | 230 | 231 | 232 | 233 | 234 | 235 | 236 | 237 | 238 | 239 |
| α | β | Γ | π | Σ | σ | µ | Τ | Φ | Θ | Ω | δ | θ | ø | Є | Π |
| 240 | 241 | 242 | 243 | 244 | 245 | 246 | 247 | 248 | 249 | 250 | 251 | 252 | 253 | 254 | 255 |
| ≡ | ± | ≥ | ≤ | ⌠ | ⌡ | ÷ | ≈ | ° | ▪ | · | √ | ⁿ | ² | ■ | |